TRAINING COMPUTER PERSONNEL
THE PROBLEM AND A POSSIBLE SOLUTION

Alfred Bork

Technical Report 232

Department of Information and Computer Science
University of California
Irvine, CA 92717

July 26, 1984

This document addresses a very serious problem; our present methods of training computer specialists, of all types, and levels, are inadequate for the task. We are producing too few experts, and projections indicate an ever greater gap between supply and demand. This shortage affects computer scientists from B. S. to Ph.D. as well as other computer personnel.

In spite of the importance of this problem, we see few adequate attempts at a solution. The problem is of national and even international concern. We can train, effectively, far more people if we use the computer itself as a major part of the training system.

THE PROBLEM

This problem of insufficient training is salient in university computer science departments. Estimates suggest that uncontrolled enrollments in computer science rise as much as 20% a year ("The Chronicle of Higher Education," February 9, 1981.) Yet these departments, besieged with students, have great problems hiring faculty. Many computer science departments, even the most well-known, have positions they are unable to fill. Many are accepting far fewer students than they would attract without restricted enrollment.

The reasons for this problem are easy to trace. First, we now graduate relatively few advanced students. Further, people who graduate from Ph.D. programs in computer science are in tremendous demand, in computer science departments, and in major industries. We do not graduate enough computer experts to satisfy demands of both industry and schools. Industrial

salaries are considerably higher than university salaries, so universities do not compete well.

Although I have described the shortage of trained individuals in computer science departments in higher education, it exists in many computer areas. We have far too few programmers, far too few analysts, far too few applications specialists, at almost every level. Furthermore, projections, are bleak; the problem is worsening. Retraining of computer specialists is also a continuing problem, particularly with new languages such as Ada.

None of the information about this crisis will be new to experts in the field, as they have been discussing this situation for years. But I find that it is not well known generally.

CURRENT ATTEMPTS AT A SOLUTION

We hear of few suggested solutions to this continuing problem, and we see little long-range planning to meet this continually increasing national crisis. Several makeshift measures prevail in universities: differential salary scales, limiting student enrollemnt, and part-time faculty. Some universities (for example, the University of California) pay more to computer science faculty, hoping to compete more favorably with industrial offers. Sometimes computer science faculty enter at higher steps in the salary scale.

But this salary differential is rare, and seldom adequate to the problem. Universities still cannot compete with high industrial salaries, and the basic difficulty--the lack of sufficient numbers of competent computer science experts--is not

addressed.

One widely needed approach is to limit the number of computer science students. But this can only compound the problem in the future.

A typical way of coping with this shortage has been to employ part-time faculty. Many schools do this already. The problem shows signs of becoming more acute, and it will soon be difficult even to hire the part-time faculty needed. Further, part-time faculty are a mixed blessing; some are competent teachers, and some are not. In some areas relatively few people are available, with standards of curriculum deteriorating as a result.

We are facing a major national problem, important for the future of this country, for which few positive solutions have been suggested. Although I have referred mostly to computer science programs in universities, similar problems exist in many other areas for training computer personnel. Again, few promising solutions have been proposed.

A POSSIBLE SOLUTION

This situation is not hopeless. I envision a promising approach for yielding many more high quality computer science courses <u>in and out</u> of the university, with faculty and other teachers already on hand. Furthermore, I argue that no other competing strategy for coping with the situation shows similar promise. We can maintain and even improve the quality of the typical learning experience.

I suggest we develop full, flexible, <u>computer based</u> courses

in computer science, for the beginning level courses and perhaps for other courses also. These completely developed courses, could require little time of teachers. Based on what we already know about the computer as a learning device, the computer could serve not only as the object of study, but as the principal vehicle for delivering learning material to students. Other modes, such as print and film, might be involved, but the computer would be the major delivery system for the learning material. Staff requirements could thus be reduced.

The computer has several major advantages in learning. First, it can provide an active learning environment for each student, difficult to do in lectures and in books. Second, it can allow students to proceed through the learning material at rates to match each student's needs. Third, it can provide an individualized learning experience for each student. Student difficulties can be quickly identified and individualized aid can be offered.

The computer has probably seen less use as a learning device within the discipline of computer science than several other disciplines! In the sciences and mathematics, a variety of computer learning materials are available, although not typically full courses. A notable exception is the extensive development of on-line quizzes in Pascal that Kenneth Bowles pioneered at the University of California, San Diego.[1] But mostly the use of the computer as a learning device has occurred in areas other than computer science itself.

The computer is already an essential component in learning

how to program, an activity which demands practice. Some learning aid from the computer is inevitable even if it comes only in the form of error messagers from a compiler. I am proposing an extension of the role of the computer.

The appearance of a new computer language, Ada, likely to be widely used, raises an interesting new set of possibilities. Many computer science departments and other groups may be moving toward the use of Ada in the future for some courses. We will need many retraining courses in Ada. Few Ada courses exist now at any level. As Ada becomes more popular with a greater and greater number of people, it may well serve as a basis for teaching structured programming or for teaching software engineering or programming environment concepts. This new language, with very few courses currently devoted to it, would furnish an interesting trial case for these ideas.

COMPUTERS AND LEARNING

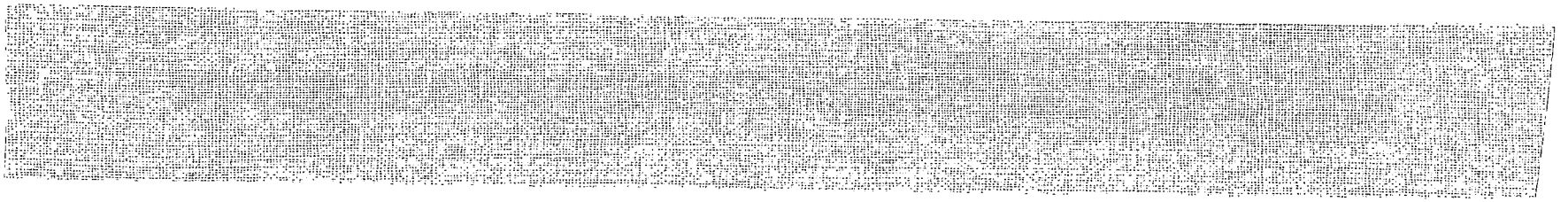
Good informal recent evidence, suggests that computers can be effective learning devices. Experience in developing such materials has grown rapidly. (It is difficult to gather formal evidence of this kind in any learning medium.) With good computer based learning material, we can create an interactive learning environment, where students are constantly asked to play active roles. In addition, we can easily and naturally handle the inevitable differences with different learners. Many examples of effective learning material exist, and a few even present whole courses.[2]

We have also demonstrated, in materials developed for the

public library,[3] that these learning units can run independent of human aid. That is, they can, if properly designed, work in situations where the teacher plays only a minor role, or no role; so the delivery of instruction is no longer the responsibility of a person. A competent teacher can still assist students with difficulties offering additional aid beyond that given by the learning materials themselves, on a one-to-one basis. The net effect of a computer based course can be a much more humane course than is possible in present large lecture-based classes.

I wish to make it clear that I am talking about very well developed curriculum material, not minor efforts by a single individual with little support. Developing of effective learning material in any medium, computer or otherwise, and in any area, demands coordinated effort. Curriculum development can only be done effectively within a careful and adequately financed project. Such an effort is needed in computer science courses.

The Open University in England[4] provides an example of good curriculum development, although often not involving much use of the computer. American universities seldom develop materials with the skill and scale of The Open University. I do not claim that all the courses of The Open University are highly successful, but the overall quality is high. The Open University studies show forcefully that the overall costs of higher education can be reduced, with quality of learning maintained or improved, if more effort is focused on careful, adequately financed development of the learning units. The major curriculum developments in the United States, mostly in the



1960's, also show the value of careful development of learning material. These same techniques can be applied to computer based learning materials.

The enterprise for creating the proposed courses would have to be well planned, well staffed and well financed if it is to produce good learning material to work effectively with a large range of individuals. It would need to include, in the pedagogical development phase, the best current teachers of computer science from all parts of this country and the world. The process for producing large-scale courses employing computer based learning is well understood.[5]

One might ask if the widespread use of computer based courses in computing would discourage people from the study of computing. Quite the contrary, it would open up computing to a much greater number of people than currently can get into our courses. Furthermore, it would offer excellent, well designed courses for students who are not able to attend the university or who cannot attend the "best" schools. Thus such use would equalize educational opportunity in the country. Our experience at Irvine, in offering the beginning physics course in both computer and non-computer form, is that students each year strongly prefer the computer form over the non-computer form.

STRUCTURE OF THE COMPUTER BASED LEARNING MATERIAL

One general instructional approach for guiding the development of the proposed courses is that of the learning cycle. The learning cycle, as developed by Robert Karplus at the University of California, Berkeley, from Piaget's developmental

approach to psychology, has three important components. The first component is an experiential component. The learner is provided with a variety of exploratory experiences to establish a context for the final learning activity. The aim is to build the learner's intuition and insight into the phenomena without formal instruction. The activities in the second stage resemble those in most curriculum developments, which often start here, having ignored the initial experiential component. The third component of the cycle, often reflected only in final testing, is to see what students can do with the material just learned. The three components can be in repeated cycles rather than being a single linear progression.

An auxiliary idea which fits in naturally with the learning cycle approach and which we could follow with computer science materials is mastery learning. The somewhat pretentious term "mastery" conveys the idea that a student should not leave a particular domain until it is clear, both to the student and the instructor, that that domain has been fully learned. Thus, no "gentleperson's pass" is possible in mastery learning.

A most important aspect of mastery learning is to provide a variety of learning materials. If a student does not learn the concepts involved from one approach, that student may need another approach, different from a conceptual point of view or from a media or presentation point of view. With the computer it is possible to realize the mastery learning situation, even with very large groups of learners.

In realizing this pedagogical approach with computer based

learning, a fundamental concept is that we can combine learning and testing, within the learning cycle and mastery learning points of view, into a single activity. This is practical only with the computer. Learning and testing have usually been viewed as quite different activities, conducted separately.

We can follow two strategies, different primarily in how they are presented to the learner rather than in terms of the material. The first of these strategies is for the learner to see, as part of the computer material, a set of explicit tests, the strategy followed in the introductory physics course[6] by the Educational Technology Center. Several tests cover each unit of that course, and students must pass these tests at the mastery level, testing several times if necessary, before proceeding. Within each test is a large amount of learning material, often highly specific to just the shortcomings the student has revealed while testing. The flow from testing to learning to testing and back is smooth and frequent; the two activities are combined.

The second strategy for combining learning and testing is to bury the tests within the learning material. Students don't explicitly see that they are tested. The entire material appears as learning material, but nevertheless testing phases are mixed with the learning phases. The material on scientific literacy for public libraries developed at the Educational Technology Center illustrates this approach.

Both these approaches are self-paced, with students having partial control over the rate at which they move through the material. Mastery learning can be fully realized only in a self-

paced situation. Students may take different amounts of time through the material; the system must allow for these differences if the principle of mastery is to be maintained.

A computer based course can also include all management capabilities. This includes both management from the students' point of view, the ability to receive large amounts of individualized feedback, and management from the instructor's point of view, i.e., the instructor's ability to locate the difficulties in the class and to act on these difficulties and to maintain class records. The exact details differ somewhat between the timesharing environment and the personal computer environment. But in all cases management capabilities are possible. As a fringe benefit, the computer handles all the records in the class, obviating skilled human attention for this clerical task, sizable in large courses.[7]

The ideas presented in this section are not new. They have served at developmental groups for many years. Thus, they are already demonstrably effective for an overall pedagogical strategy for curriculum development. Many other aspects of curriculum development must also be considered in the overall project.

PROPOSED COURSES

The following brief list indicates some of the courses which may be developed. It will serve as a starting point for further discussion. Many other courses could be proposed.

It is not necessary to develop all these courses a single project. Indeed, each of the courses might be a project in its

own right. Nevertheless, some common material can be shared, particularly with the first three.

1. Introductory University Level Course

This course would replace the introductory computer science course in colleges and universities. We could allow some language variants. The stress would still be on structured programming ideas, and on programming style. We would concentrate on preparing students for later undergraduate courses in the university.

2. Introduction to Ada - Nonuniversity Course

This course would assume no previous acquaintance with computer programming. It would focus strongly on the structured programming ideas, introducing Ada within a series of examples which stress how to develop programs. Optional units of the course might stress different areas in which the programmer would intend to work eventually. So the course would, for any one student, have a practical flavor very much oriented toward the programming tasks that individual would expect to do later. However, much of the material would be common to all students.

3. Retraining Course, from Some Other Programming Language to Ada

This course would not be primarily a university course, although it might be useful for some beginning students in universities. It would be intended for those already familiar with an older programming language. The goal would be to help them make the transition to the use of Ada.

The course would very strongly emphasize the ideas of

structured programming, programming style, and an effective programming environment. These ideas (rather than the grammatical details) are likely to cause the greatest problem to experienced programmers in making the transition to Ada. Considerable emphasis could be placed on some of the new features of Ada not typically present in older languages, such as packages, separate compilation, strong typing, generics, concurrent processes, and exception handling.

4. Introduction to Computers - Second Level

This course would be a typical second computer science course for universities.

5. Introduction to Data Structures

This course would introduce the types of data structures common to computer science.

6. Applications of Computers

Courses could be developed for different applications.

I emphasize again that these courses are for illustration, and that they might not all be part of the same curriculum development project. Courses two, three, and six are in a different group from the others in terms of their intended usage, referring to a nonuniversity audience primarily. Courses one, four and five are the foundation of a computer science curriculum at a college or university. Different groups might develop different courses.

PRODUCTION

The production strategies for these materials might be those developed and extensively used at the Educational Technology

Center during the past sixteen years.[5] Probably several courses of each type should be developed to allow variety.

A developmental project would begin with an internal staff review of the literature, to identify the best ways of learning the course or courses under discussion. Next would come a position paper, referencing this literature and describing the tactics selected. We would consider not only conventional ways of teaching programming but also some newer ways based on intelligent editors and other programming aids.[8]

This study would be followed by a meeting of experts from all over the country to outline the course strategies. Developers need to exercise their imaginations to consider new structures possible with computers. This meeting would produce a detailed course outline for each unit. Each of the writing teams in the next phase would receive these unit outlines.

In the main stage of pedagogical design, writing groups working closely together would prepare the detailed specifications of each unit. For many years the Educational Technology Center has used a script, a modified flowchart form, to produce hundreds of hours of material. A group of three or four working for a week, will generally produce about one to two hours of material for students. It is critical to involve excellent experienced teachers, wherever they may live.

The next phase would be visual design of the material, the design of the screen, by competent graphic designers. Many projects neglect this important stage. Special design software is available to help the designers.

The next stage would be coding, possibly with a combination of carefully trained undergraduate student programmers and professional coders. To make the material transferable to various delivery machines and to assure that successive revisions of the material are easy, the material would be written following the best tactics of modern structured programming and software engineering. Research is needed to make the process more efficient.

For practical purposes, particularly where costs must be minimized, the delivery machines will likely be the new generation of 16-bit and 32-bit personal computers. But development hardware will be more complex.

Review, evaluation, and improvement stages should be included at several steps in the development process. Of particular importance are several cycles of formative evaluation and improvement involving the target student audience of a course. The computer, through its information saving capability, can aid in this processl.

Distribution of the final course would probably be through commercial sources, textbook publishers, computer vendors, or new companies formed for this prupose. Possibilities for non-commercial distribution should also be considered.

References

1. Microcomputer Based Mass Education, Kenneth Bowles, Institute for Information Systems, University of California, San Diego, 1977.

Problem Solving Using UCSD Pascal, Second Edition, Bowles, Kenneth L., Franklin, Stephen D., Volper, Dennis J.,

Springer-Verlag, New York, 1984.

2. Learning with Computers, Alfred Bork, Digital Press, Billerica, Massachusetts, 1981.

Personal Computers for Education, Alfred Bork, Harper and Row, due 1984.

3. Learning with Computers in Public Enviroments, Alfred Bork, July 1981.

Science Literacy in the Public Library--Batteries and Bulbs, Arnold Arons, Alfred Bork, Frank Collea, Stephen Franklin, Barry Kurtz, Proceedings of National Educational Computing Conference 1981.

Newton--A Mechanical World, Alfred Bork, Stephen Franklin, Martin Katz, John McNelly, Proceedings of National Educational Computing Conference 1981.

4. The Open University, Walter Perry (San Francisco: Jossey-Bass, 1977).

5. Production Systems for Computer Based Learning, Alfred Bork, February 1982. Prepared for the Department of Education. Reprinted in Instructional Software: Principles and Perspectives for Design and Use, Decker F. Walker and Robert D. Hess, Editors, Wadsworth Publishing Company, Belmont, California, 1984.

6. Educational Technology Center at the University of California, Irvine, IFIPS, Summer 1979.

Modes of Computer Usage in Physics (with Joseph Marasco), THE Journal, October 1976.

Computer-based Instruction in Physics, Physics Today, Vol. 34, No. 9, September 1981.

7. Course Management System for the Physics 3 Course at Irvine, Alfred Bork, Joseph Marasco and Jeff East, in Learning with Computers, Digital Press, 1981.

8. The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, Tim Teitelbaum and Thomas Reps, Communications of the ACM, September 1981, Volume 24, page 563-573.